S90.09.exam.21q

Number: S90.09
Passing Score: 800
Time Limit: 120 min

S90.09

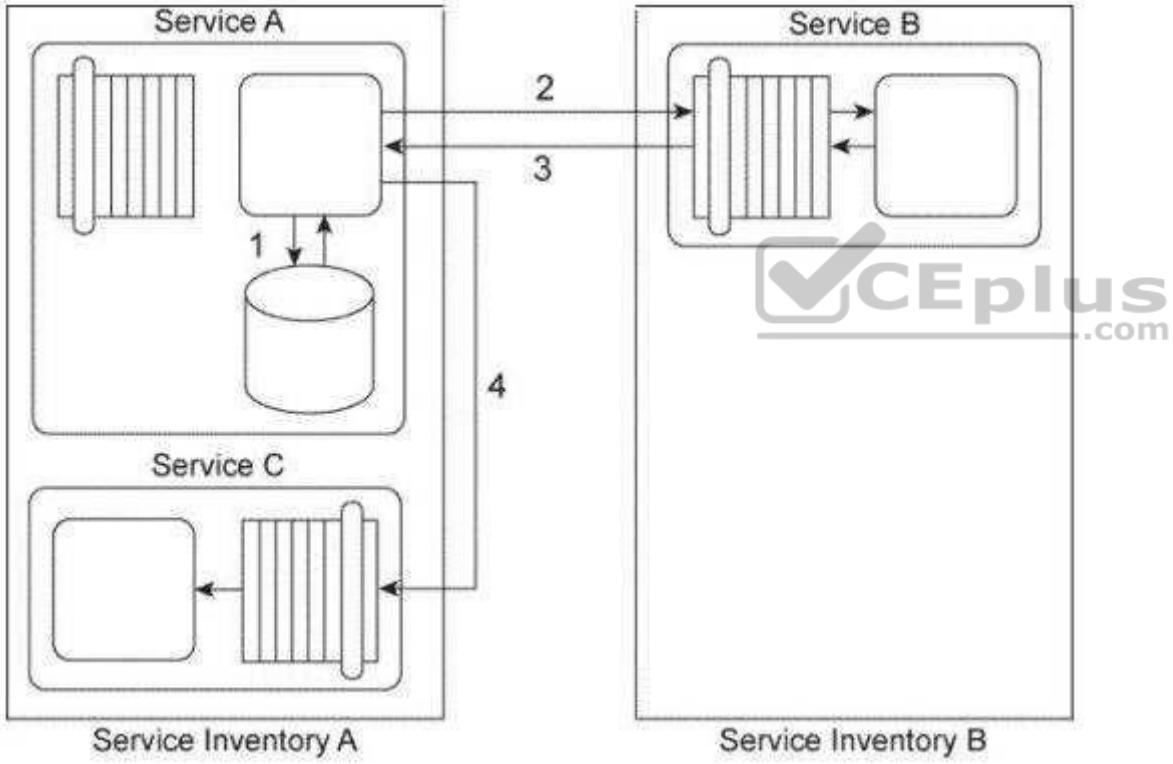
SOA Design & Architecture Lab

Exam A

QUESTION 1
Service A is a task service that sends Service B a message (2) requesting that Service B return data back to Service A in a response message (3). Depending on the response received. Service A may be required to send a message to Service C (4) for which it requires no response.

Before it contacts Service B, Service A must first retrieve a list of code values from its own database (1) and then place this data into its own memory. If it turns out that it must send a message to Service C, then Service A must combine the data it receives from Service B with the data from the code value list in order to create the message it sends to Service C. If Service A is not required to invoke Service C, it can complete its task by discarding the code values.

Service A and Service C reside in Service Inventory A. Service B resides in Service Inventory B.



You are told that the services in Service Inventory A are all SOAP-based Web services designed to exchange SOAP 1.1 messages and the services in Service Inventory B are SOAP-based Web services designed to exchange SOAP 1.2 messages. Therefore, Service A and Service B cannot currently communicate.

Furthermore, you are told that Service B needs to access a shared database in order to retrieve the data required by Service A. The response time of the database can sometimes be lengthy, which would cause Service A to consume too much resources while it is waiting and keeping the code values in memory. How can this service composition architecture be changed to avoid these problems?

A. The Protocol Bridging pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can convert SOAP 1.1 messages to SOAP 1.2 messages and vice versa. The Service Data Replication pattern can be applied to Service B so that it is given a dedicated database with its own copy of the data it needs to access. The Service Normalization pattern can then be applied to ensure that the data within the replicated database is normalized with the shared database it is receiving replicated data from.

B. The Protocol Bridging pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can convert SOAP 1.1 messages to SOAP 1.2 messages and vice versa. The Service Statelessness principle can be applied with the help of the State Repository pattern so that Service A can write the code value data to a state database while it is waiting for Service B to respond.

C. The Protocol Bridging pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can convert SOAP 1.1 messages to SOAP 1.2 messages and vice versa. The Intermediate Routing pattern can be applied to dynamically determine whether Service A should send a message to Service C. The Service Autonomy principle can be applied to Service A to further increase its behavioral predictability by reducing the amount of memory it is required to consume. D. None of the above.

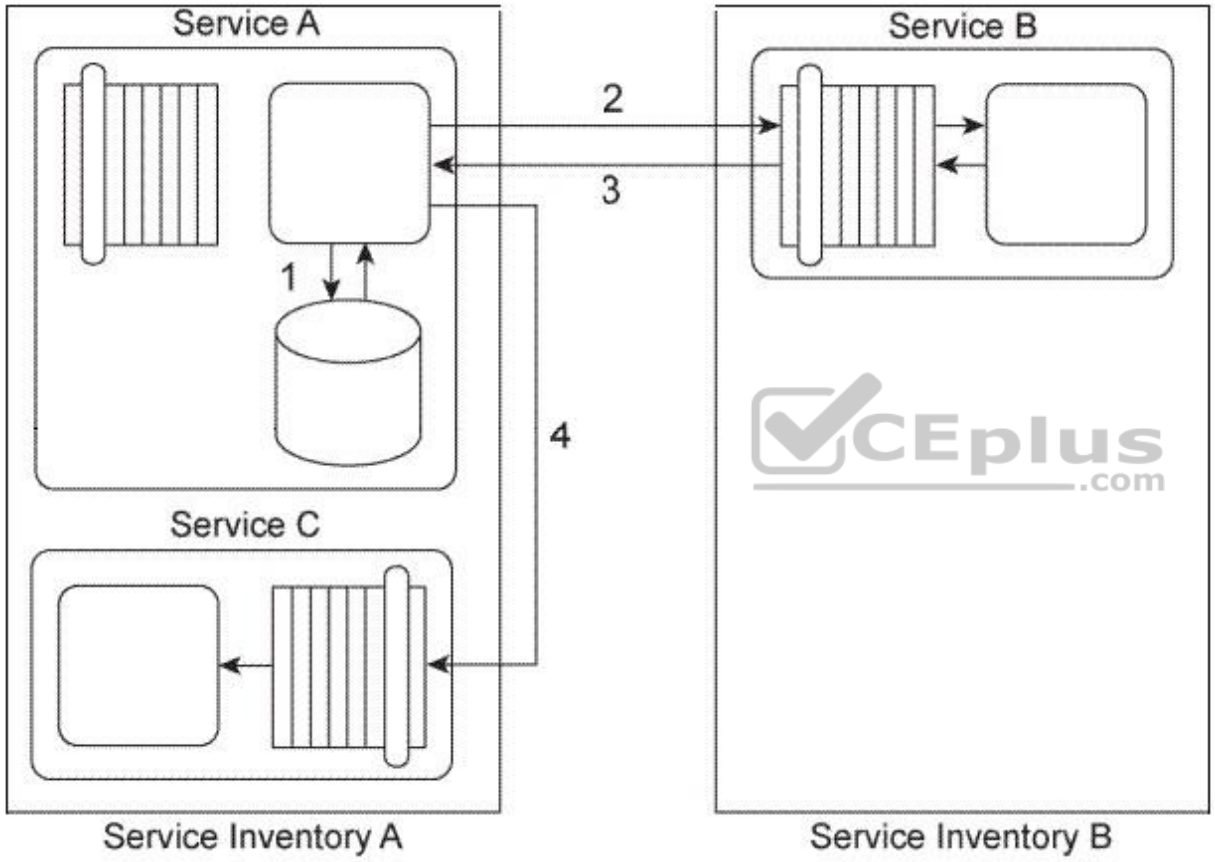
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 2
Service A is a task service that sends Service B a message (2) requesting that Service B return data back to Service A in a response message (3). Depending on the response received. Service A may be required to send a message to Service C (4) for which it requires no response.

Before it contacts Service B, Service A must first retrieve a list of code values from its own database (1) and then place this data into its own memory. If it turns out that it must send a message to Service C, then Service A must combine the data it receives from Service B with the data from the code value list in order to create the message it sends to Service C. If Service A is not required to invoke Service C, it can complete its task by discarding the code values.

Service A and Service C reside in Service Inventory A. Service B resides in Service Inventory B.



You are told that the services in Service Inventory A were designed with service contracts based on different design standards than the services in Service Inventory B. As a result, Service A and Service B use different data models to represent the data they need to exchange. Therefore, Service A and Service B cannot currently communicate. Furthermore, Service C is an agnostic service that is heavily accessed by many concurrent service consumers. Service C frequently reaches its usage thresholds during which it is not available and messages sent to it are not received. How can this service composition architecture be changed to avoid these problems?

A. The Data Model Transformation pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can transform a message from one data model to another at runtime. The Intermediate Routing and Service Agent patterns can be applied so that when Service B sends a response message, a service agent can intercept the message and, based on its contents, either forward the message to Service A or route the message to Service C . The Service Autonomy principle can be further applied to Service C together with the Redundant Implementation pattern to help establish a more reliable and scalable service architecture.

B. The Data Model Transformation pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can transform a message from one data model to another at runtime. The Asynchronous Queuing pattern can be applied to establish an intermediate queue between Service A and Service C so that when Service A needs to send a message to Service C, the queue will store the message and retransmit it to Service C until it is successfully delivered. The Service Autonomy principle can be further applied to Service C together with the Redundant Implementation pattern to help establish a more reliable and scalable service architecture.

C. The Data Model Transformation pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can transform a message from one data model to another at runtime. The Intermediate Routing and Service Agent patterns can be applied so that when Service B sends a response message, a service agent can intercept the message and, based on its contents, either forward the message to Service A or route the message to Service C . The Service Statelessness principle can be applied with the help of the State Repository pattern so that Service A can write the code value data to a state database while it is waiting for Service B to respond. D. None of the above.

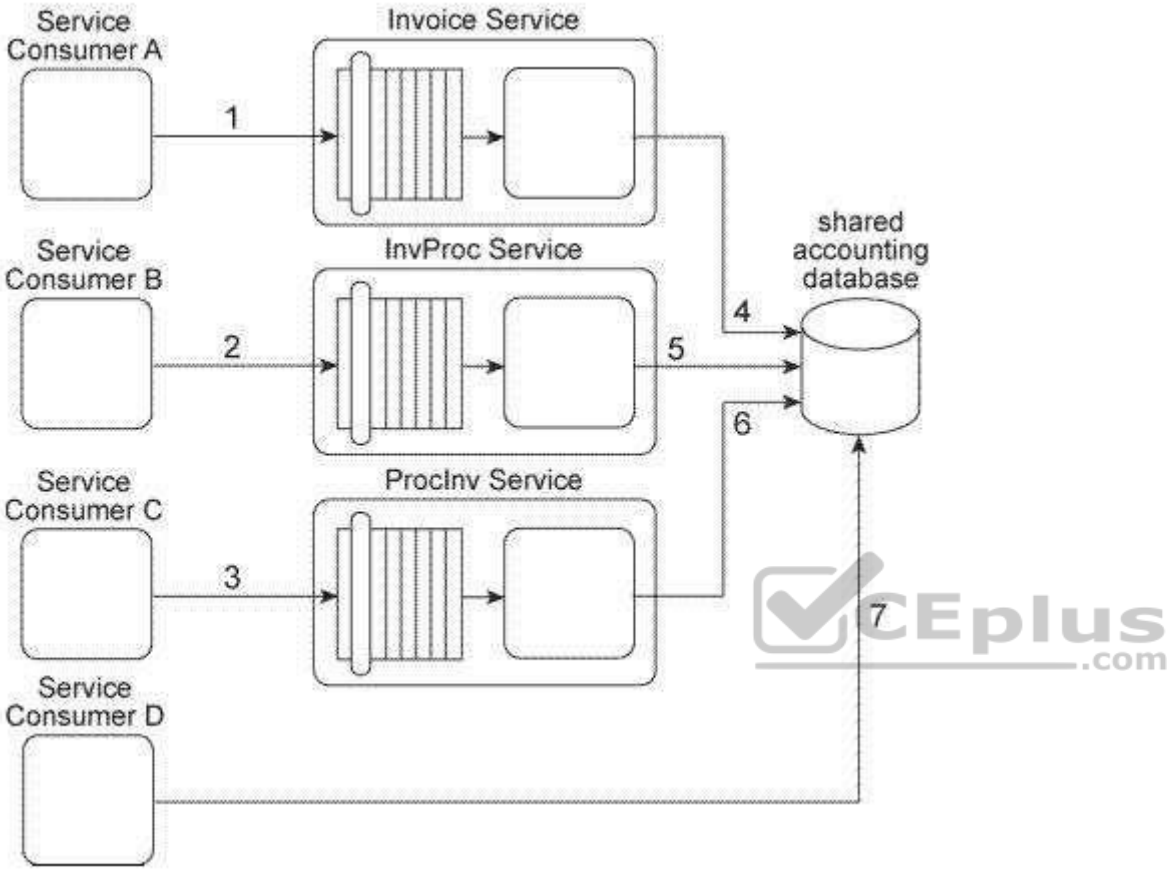
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 3
Our service inventory contains the following three services that provide invoice-related data access capabilities: Invoice, InvProc, and Proclnv. These services were created at different times by different project teams and were not required to comply to any design standards. Therefore each of these services has a different data model for representing invoice data.

Currently each of these three services has one service consumer: Service Consumer A accesses the Invoice service(1). Service Consumer B (2) accesses the InvProc service, and Service Consumer C (3) accesses the Proclnv service. Each service consumer invokes a data access capability of an invoice-related service, requiring that service to interact with the shared accounting database that is used by all invoice-related services (4, 5, 6).

Additionally, Service Consumer D was designed to access invoice data from the shared accounting database directly (7), (Within the context of this architecture. Service Consumer D is labeled as a service consumer because it is accessing a resource that is related to the illustrated service architectures.)



Assuming that the Invoice service, InvProc service, and ProcInv service are part of the same service inventory, what steps would be required to fully apply the Official Endpoint pattern?

A. One of the invoice-related services needs to be chosen as the official service providing invoice data access capabilities. Service Consumers A, B, and C then need to be redesigned to only access the chosen invoice-related service. Because Service Consumer D does not rely on an invoice-related service, it is not affected by the Official Endpoint pattern and can continue to access the accounting database directly. The Service Abstraction principle can be further applied to hide the existence of the shared accounting database and other implementation details from current and future service consumers.

B. One of the invoice-related services needs to be chosen as the official service providing invoice data access capabilities. Service Consumers A, B, and C then need to be redesigned to only access the chosen invoice-related service. Service Consumer D also needs to be redesigned to not access the shared

accounting database directly, but to also perform its data access by interacting with the official invoice-related service. The Service Abstraction principle can be further

applied to hide the existence of the shared accounting database and other implementation details from current and future service consumers.

C. Because Service Consumers A, B, and C are already carrying out their data access via published contracts, they are not affected by the Official Endpoint pattern. Service Consumer D needs to be redesigned to not access the shared accounting database directly, but to perform its data access by interacting with the official invoice-related service. The Service Abstraction principle can be further applied to hide the existence of the shared accounting database and other implementation details from current and future service consumers. D. None of the above.

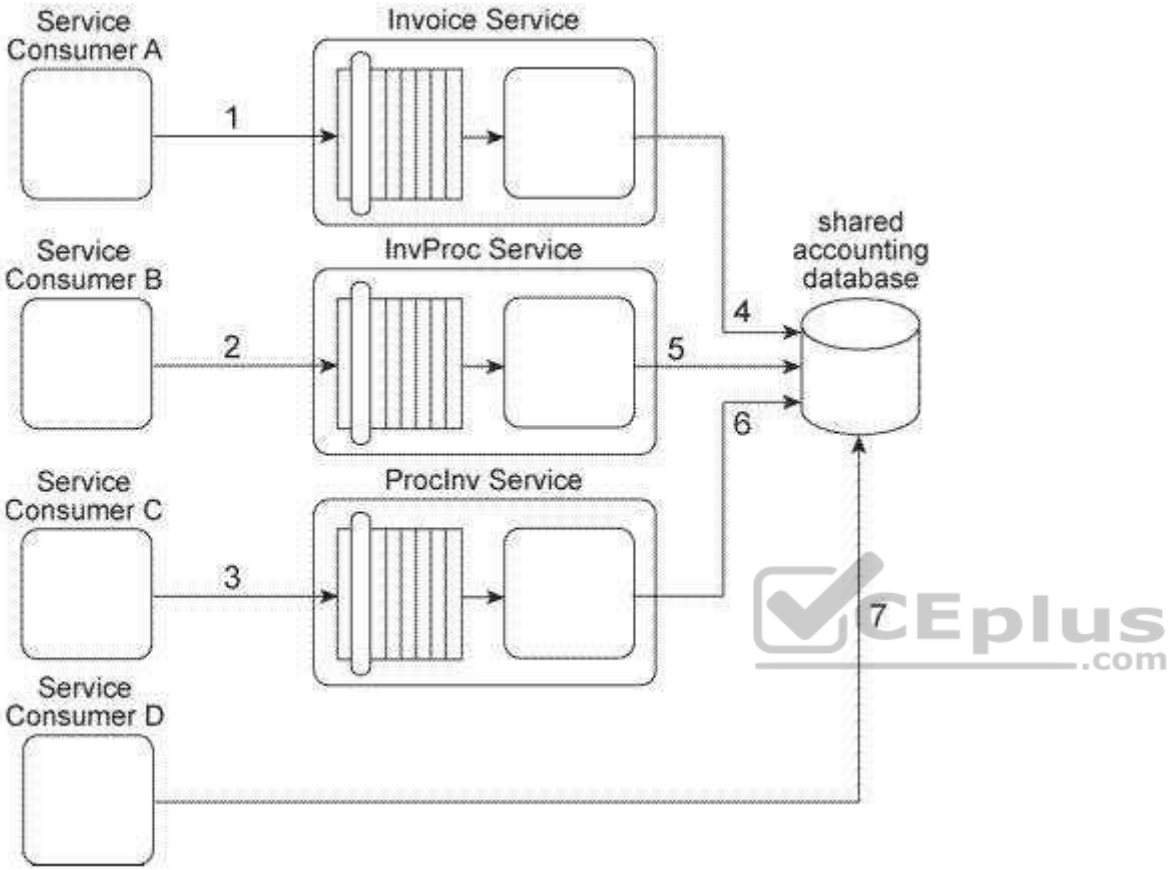
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 4
Our service inventory contains the following three services that provide invoice-related data access capabilities: Invoice, InvProc, and Proclnv. These services were created at different times by different project teams and were not required to comply to any design standards. Therefore each of these services has a different data model for representing invoice data.

Currently each of these three services has one service consumer: Service Consumer A accesses the Invoice service(1). Service Consumer B (2) accesses the InvProc service, and Service Consumer C (3) accesses the Proclnv service. Each service consumer invokes a data access capability of an invoice-related service, requiring that service to interact with the shared accounting database that is used by all invoice-related services (4, 5, 6).

Additionally, Service Consumer D was designed to access invoice data from the shared accounting database directly (7). (Within the context of this architecture. Service Consumer D is labeled as a service consumer because it is accessing a resource that is related to the illustrated service architectures.)



A project team recently proclaimed that it has successfully applied the Contract Centralization pattern to the service inventory in which the Invoice service, InvProc service, and ProcInv service reside. Upon reviewing the previously described architecture you have doubts that this is true. After voicing your doubts to a manager, you are asked to provide specific evidence as to why the Contract Centralization is not currently fully applied. Which of the following statements provides this evidence?

A. The Contract Centralization pattern is not fully applied to the Invoice, InvProc, and ProcInv services because they are being accessed by different service consumers and because they have redundant logic that introduces denormalization into the service inventory.
B. The Contract Centralization pattern is not fully applied because Service Consumer D is accessing the shared accounting database directly.

C. The Contract Centralization pattern is not fully applied because none of the invoice-related services are carrying out data access logic via a centralized and standardized invoice service. This is primarily because the Standardized Service Contract principle was not consistently applied during the delivery processes of

the individual services.
D. None of the above.

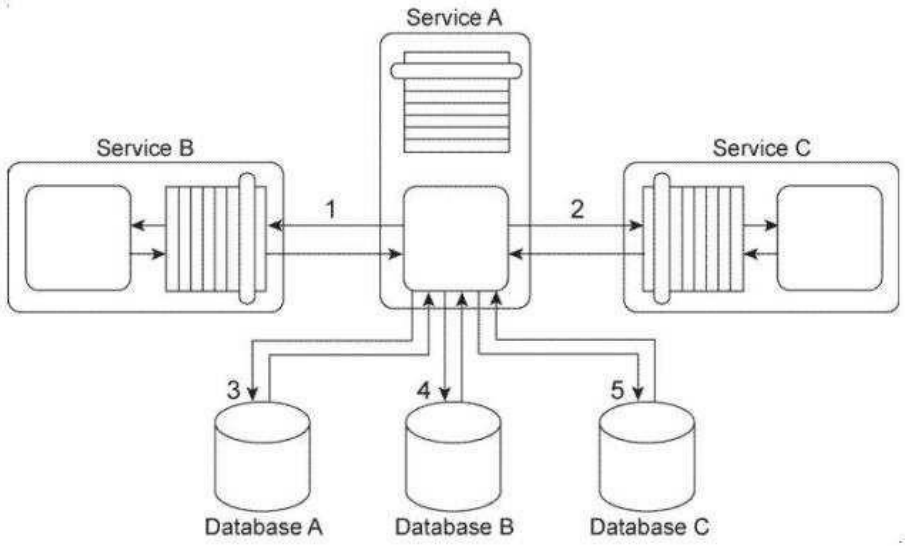
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 5
Service A is an entity service that provides a set of generic and reusable service capabilities. In order to carry out the functionality of any one of its service capabilities, Service A is required to compose Service B (1) and Service C (2) and Service A is required to access Database A (3), Database B (4), and Database C (5). These three databases are shared by other applications within the IT enterprise.

All of service capabilities provided by Service A are synchronous, which means that for each request a service consumer makes. Service A is required to issue a response message after all of the processing has completed.

Depending on the nature of the service consumer request, Service A may be required to hold data it receives in memory until its underlying processing completes. This includes data it may receive from either Service A or Service B or from any of the three shared databases.
Service A is one of many entity services that reside in a highly normalized service inventory. Because Service A provides agnostic logic, it is heavily reused and is currently part of many service compositions.



You are told that Service A has recently become unstable and unreliable and several of the service consumers that access it have had to raise runtime exceptions due to these problems. What steps can be taken to solve these problems without compromising the normalization of the service inventory?

A. The Service Autonomy principle can be applied to increase the physical isolation of Service A and to reduce dependencies Service A has on external resources. In support of this, the Service Data Replication pattern can be applied in order to establish a dedicated database that contains replicated data from shared Databases A, B, and C . Furthermore, the Redundant Implementation pattern can be applied so that the logic Service A requires from Services B and C can be redundantly placed inside of Service A . This way, Service A avoids having to separately compose Services B and C

B. The Service Statelessness principle can be applied with the help of the State Repository pattern in order to establish a state database that Service A can use to defer state data it may be required to hold for extended periods. The Service Autonomy principle can also be applied in order to increase the physical isolation of Service A and to reduce dependencies Service A has on external resources. In support of this, the Service Data Replication pattern can be applied in order to establish a dedicated database that contains replicated data from shared Databases A, B, and C.

C. The Service Loose Coupling and Standardized Service Contract principles can be applied by introducing a separate utility service that provides centralized data access to the Databases A, B, and C, and exposes a standardized service contract that can be used by Service A . This will prevent Service A from direct dependencies on the shared databases in case any of them are replaced in the future. By following this approach, the Legacy Wrapper pattern is effectively applied via the introduction of the new utility service. D. None of the above.

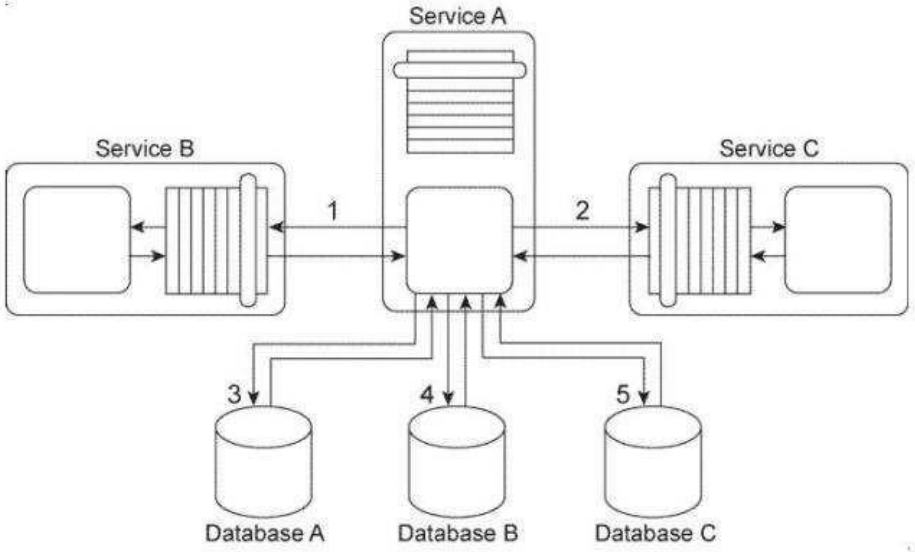
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 6
Service A is an entity service that provides a set of generic and reusable service capabilities. In order to carry out the functionality of any one of its service capabilities, Service A is required to compose Service B (1) and Service C (2) and Service A is required to access Database A (3), Database B (4), and Database C (5). These three databases are shared by other applications within the IT enterprise.

All of service capabilities provided by Service A are synchronous, which means that for each request a service consumer makes. Service A is required to issue a response message after all of the processing has completed.
Depending on the nature of the service consumer request, Service A may be required to hold data it receives in memory until its underlying processing completes. This includes data it may receive from either Service A or Service B or from any of the three shared databases.

Service A is one of many entity services that reside in a highly normalized service inventory. Because Service A provides agnostic logic, it is heavily reused and is currently part of many service compositions.



You are told that Service A has recently become unstable and unreliable. The problem has been traced to two issues with the current service architecture. First, Service B, which is also an entity service, is being increasingly reused and has itself become unstable and unreliable. When Service B fails, the failure is carried over to Service A . Secondly, shared Database B has a complex data model. Some of the queries issued by Service A to shared Database B can take a very long time to complete. What steps can be taken to solve these problems without compromising the normalization of the service inventory?

A. The Redundant Implementation pattern can be applied to Service A, thereby making duplicate deployments of the service available. This way, when one implementation of Service A is too busy, another implementation can be accessed by service consumers instead. The Service Data Replication pattern can be applied to establish a dedicated database that contains an exact copy of the data from shared Database B that is required by Service A .

B. The Redundant Implementation pattern can be applied to Service B, thereby making duplicate deployments of the service available. This way, when one implementation of Service B is too busy, another implementation can be accessed by Service A instead. The Service Data Replication pattern can be applied to establish a dedicated database that contains an exact copy of the data from shared Database B that is required by Service A .

C. The Redundant Implementation pattern can be applied to Service B, thereby making duplicate deployments of the service available. This way, when one implementation of Service B is too busy, another implementation can be accessed by Service A instead. The Service Data Replication pattern can be applied to establish a dedicated database that contains a copy of the data from shared Database B that is required by Service A . The replicated database is designed with an optimized data model in order to improve query execution performance. D. None of the above.

Correct Answer: C
Section: (none)
Explanation

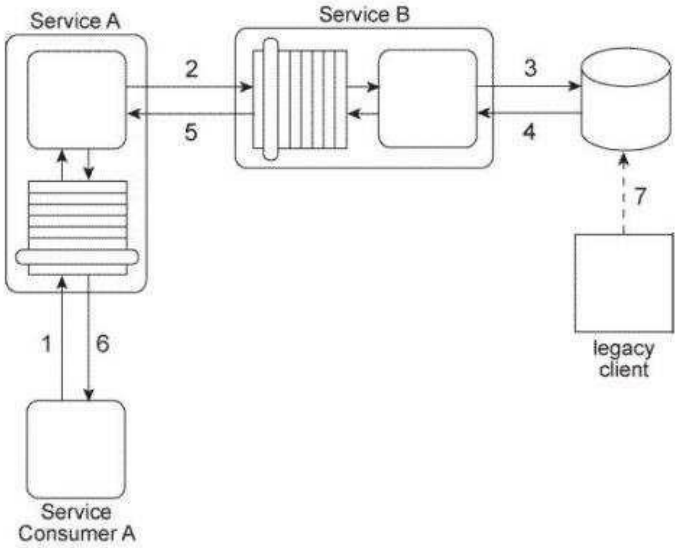
Explanation/Reference:

QUESTION 7
Service A is an entity service that provides a Get capability that returns a data value that is frequently changed.

Service Consumer A invokes Service A in order to request this data value (1). For Service A to carry out this request, it must invoke Service B (2), a utility service that interacts (3.4) with the database in which the data value is stored. Regardless of whether the data value changed, Service B returns the latest value to Service A (5), and Service A returns the latest value to Service Consumer A (6).

The data value is changed when the legacy client program updates the database (7) When this change happens is not predictable. Note also that Service A and Service B are not always available at the same time.

Any time the data value changes. Service Consumer A needs to receive it as soon as possible. Therefore, Service Consumer A initiates the message exchange shown in the Figure several times a day. When it receives the same data value as before, the response from Service A is ignored. When Service A provides an updated data value, Service Consumer A can process it to carry out its task.



The current service composition architecture is using up too many resources due to the repeated invocation of Service A by Service Consumer A and the resulting message exchanges that occur with each invocation. What steps can be taken to solve this problem?

A. The Event-Driven Messaging pattern can be applied by establishing a subscriber-publisher relationship between Service A and Service B . This way, every time the data value is updated, an event is triggered and Service B, acting as the publisher, can notify Service A, which acts as the subscriber. The

Asynchronous Queuing pattern can be applied between Service A and Service B so that the event notification message sent out by Service B will be received by Service A, even when Service A is unavailable.

B. The Event-Driven Messaging pattern can be applied by establishing a subscriber-publisher relationship between Service Consumer A and Service A . This way, every time the data value is updated, an event is triggered and Service A, acting as the publisher, can notify Service Consumer A, which acts as the subscriber. The Asynchronous Queuing pattern can be applied between Service Consumer A and Service A so that the event notification message sent out by Service A will be received by Service Consumer A, even when Service Consumer A is unavailable.

C. The Asynchronous Queuing pattern can be applied so that messaging queues are established between Service A and Service B and between Service Consumer A and Service A . This way, messages are never lost due to the unavailability of Service A or Service B. D. None of the above.

Correct Answer: D
Section: (none)
Explanation

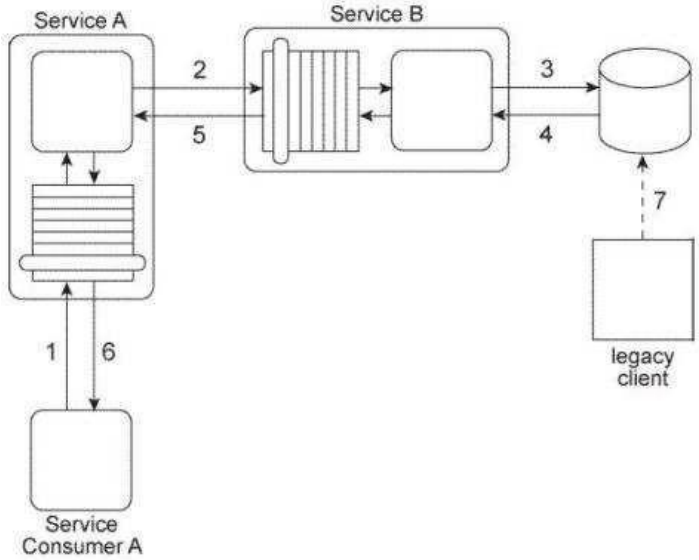
Explanation/Reference:

QUESTION 8
Service A is an entity service that provides a Get capability that returns a data value that is frequently changed.

Service Consumer A invokes Service A in order to request this data value (1). For Service A to carry out this request, it must invoke Service B (2), a utility service that interacts (3.4) with the database in which the data value is stored, Regardless of whether the data value changed. Service B returns the latest value to Service A (5), and Service A returns the latest value to Service Consumer A (6).

The data value is changed when the legacy client program updates the database (7). When this change happens is not predictable. Note also that Service A and Service B are not always available at the same time.

Any time the data value changes. Service Consumer A needs to receive it as soon as possible. Therefore, Service Consumer A initiates the message exchange shown in the Figure several times a day. When it receives the same data value as before, the response from Service A is ignored. When Service A provides an updated data value, Service Consumer A can process it to carry out its task.



Because Service A and Service B are not always available at the same times, messages are getting lost and several invocation attempts by Service Consumer A fail. What steps can be taken to solve this problem?

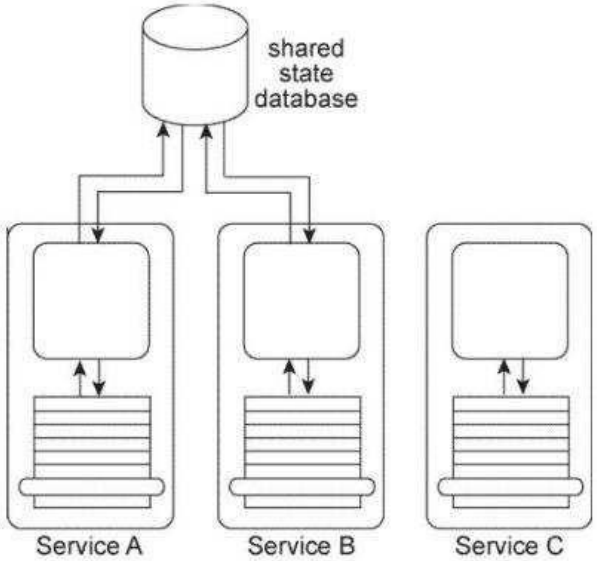
A. The Asynchronous Queuing pattern can be applied so that messaging queues are established between Service A and Service B and between Service Consumer A and Service A . This way, messages are never lost due to the unavailability of Service A or Service B .
B. The Asynchronous Queuing pattern can be applied so that a messaging queue is established between Service A and Service B . This way, messages are never lost due to the unavailability of Service A or Service B . The Service Agent pattern can be further applied to establish a service agent that makes a log entry and issues a notification when re-transmission attempts by the messaging queue exceeds a pre-determined quantity.
C. The Asynchronous Queuing pattern can be applied so that a messaging queue is established between Service Consumer A and Service A. This way, messages are never lost due to the unavailability of Service A or Service B. The Service Agent pattern can be further applied to establish a service agent that makes a log entry each time a runtime exception occurs. D. None of the above.

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:
QUESTION 9
Services A, B, and C are non-agnostic task services. Service A and Service B use the same shared state database to defer their state data at runtime.

An assessment of these three services reveals that each contains some agnostic logic, but because it is bundled together with the non-agnostic logic, the agnostic logic cannot be made available for reuse.

The assessment also determines that because Service A and Service B and the shared state database are each located in physically separate environments, the remote communication required for Service A and Service B to interact with the shared state database is causing an unreasonable decrease in runtime performance.



You are asked to redesign this architecture in order to increase the opportunity for agnostic service logic to be reused and in order to decrease the runtime processing demands so that performance can be improved. What steps can be taken to achieve these goals?

A. The Enterprise Service Bus pattern can be applied to establish an environment whereby the Process Abstraction and Process Centralization patterns are naturally applied, resulting in a clean separation of non-agnostic task services from newly designed agnostic services that are further shaped into reusable services by the application of the Service Reusability principle.
B. The Process Centralization pattern can be applied, resulting in a redesign effort where agnostic logic is removed from the three task services so that they only encapsulate non-agnostic logic. The agnostic logic is then moved to one or more new agnostic services that are shaped into reusable services by the application of the Service Reusability principle. The Process Abstraction pattern is then applied to the redesigned task services Service A and Service B, so that their logic is physically centralized, turning them into orchestrated task services.
C. The Process Abstraction pattern can be applied, resulting in a redesign effort where agnostic logic is removed from the three task services so that they only encapsulate non-agnostic logic. The agnostic logic is then moved to one or more new agnostic services that are shaped into reusable services by the application of the Service Reusability principle. The Orchestration pattern can be further applied to establish an environment whereby the Process

Centralization pattern is naturally applied to Services A and B and the State Repository pattern in naturally applied to further help avoid remote communication by providing a local and centralized state database that can be shared by both services. D. None of the above.

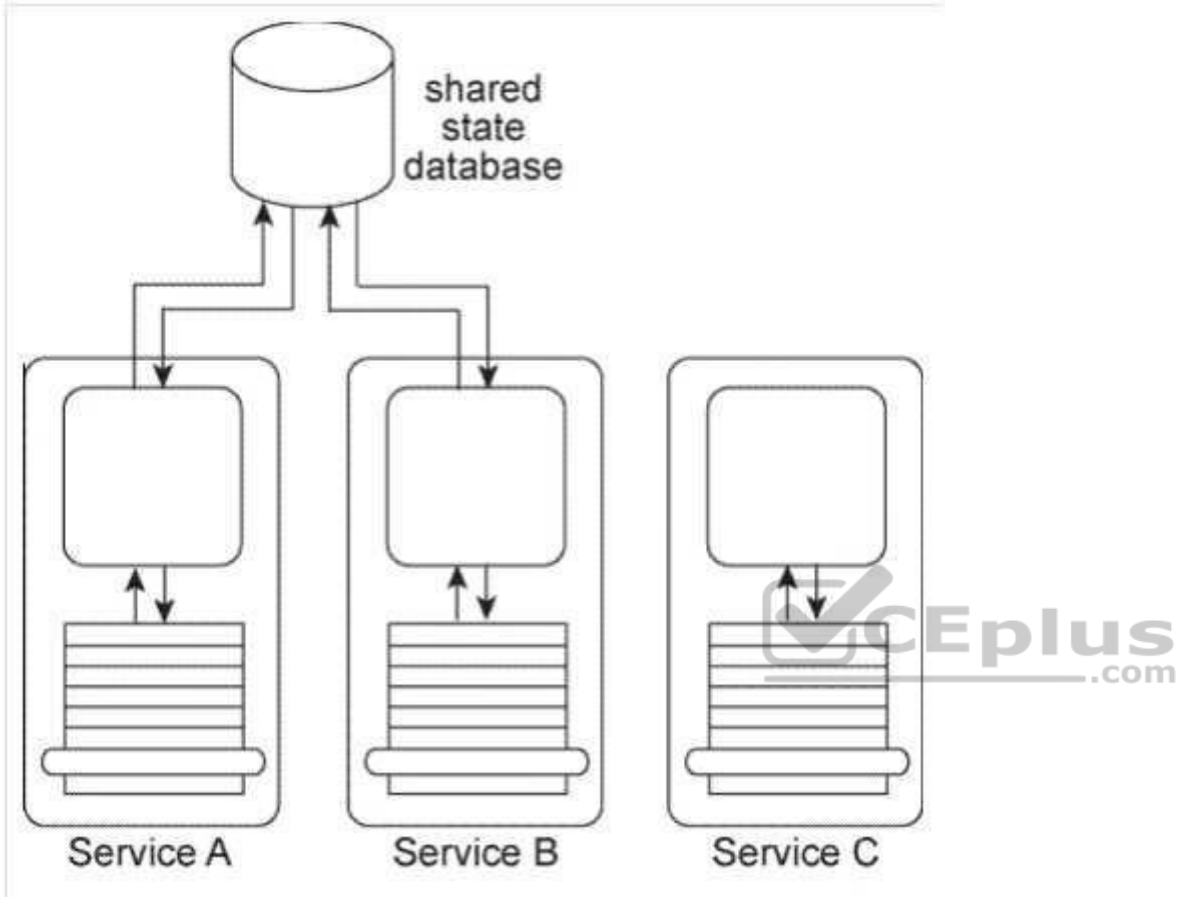
Correct Answer: C
Section: (none)
Explanation

Explanation/Reference:

QUESTION 10
Services A, B, and C are non-agnostic task services. Service A and Service B use the same shared state database to defer their state data at runtime.

An assessment of these three services reveals that each contains some agnostic logic, but because it is bundled together with the non-agnostic logic, the agnostic logic cannot be made available for reuse.

The assessment also determines that because Service A and Service B and the shared state database are each located in physically separate environments, the remote communication required for Service A and Service B to interact with the shared state database is causing an unreasonable decrease in runtime performance.



How can the application of the Orchestration pattern improve this architecture?

A. The application of the Orchestration pattern will result in an environment whereby the State Repository and Service Data Replication patterns are naturally applied, allowing the shared state database to be replicated for Services A and B so that each task service can have its own dedicated state database. The Process Centralization pattern can also be applied to Services A and B, so that their logic is physically centralized, turning them into orchestrated task services.

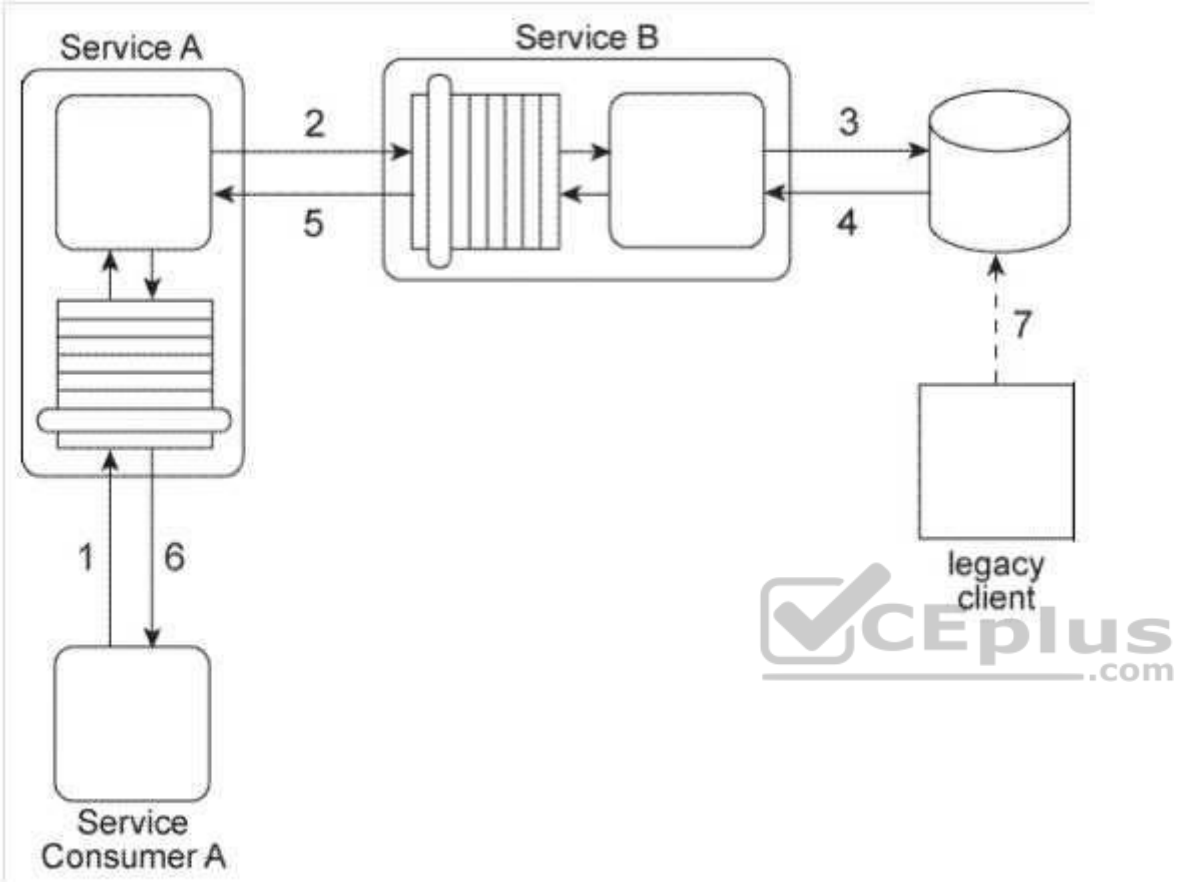
B. The application of the Orchestration pattern will result in an environment whereby the Process Abstraction and Process Centralization patterns are naturally applied to Services A, B, and C, resulting in a clean separation of non-agnostic task services from newly designed agnostic services with reuse potential. Also, the State Repository pattern can be applied by the availability of a central state database that can be shared by Services A and B. This database can be made available as a local part of the environment so that Services A and B can avoid remote communication.

C. The application of the Orchestration pattern will result in an environment whereby the Compensating Service Transaction is naturally applied, resulting in the opportunity to create sophisticated exception logic that can be used to compensate for the performance problems caused by Services A and B having to remotely access the state database. The Process Abstraction and Service Broker patterns are also naturally applied, enabling the separation of non-agnostic logic and agnostic logic while providing common transformation functions required to overcome any disparity in the service contracts that will need to be created for the new agnostic services. D. None of the above.

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 11
Service A has become increasingly difficult to maintain. Its core service logic has become bloated and convoluted because it has been updated numerous times during which additional functionality was added to interact with the database and the legacy system and to support interaction with Service Consumers A and B (via the two service contracts) as well as interaction directly with Service Consumer C.



What steps can be taken to solve these problems and to prevent them from happening again in the future?

A. The Service Facade pattern can be applied to position a Facade component between the core service logic and the implementation resources (the database and the legacy system) and to also position a Facade component between the two service contracts and Service Consumers A and B. The Official Endpoint pattern can be applied to limit access to Service A to one of its two published service contracts. The Service Loose Coupling principle can be applied so that Service Consumer C does not negatively couple itself directly to the core service logic of Service A .

B. The Service Facade pattern can be applied to position a Facade component between the core service logic and the implementation resources (the database and the legacy system) and to position a façade component between the core service logic and the two service contracts. The Contract Centralization pattern can be applied to limit access to Service A to one of its two published service contracts. The Service Abstraction principle can be applied to hide the implementation details of Service A from service consumers.

C. The Service Façade pattern can be applied to position a Facade component between the core service logic and the two service contracts. The Contract Centralization pattern can be applied to limit access to Service A to one of its two published service contracts. The Service Loose Coupling principle can be applied so that Service Consumer C does not negatively couple itself directly to the core service logic of Service A . D. None of the above.

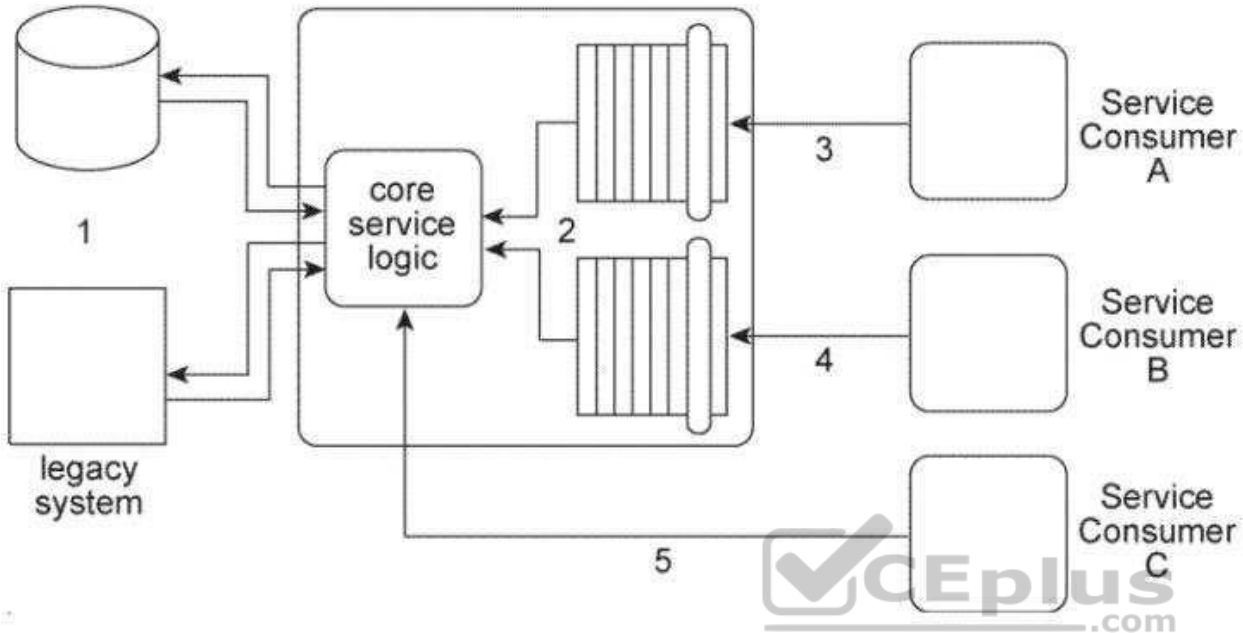
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 12
The architecture for Service A displayed in the Figure shows how the core logic of Service A has expanded over time to connect to a database and a proprietary legacy system (1) and to support two separate service contracts (2) that are accessed by different service consumers.

The service contracts are fully decoupled from the service logic. The service logic is therefore coupled to the service contracts and to the underlying implementation resources (the database and the legacy system).

Service A currently has three service consumers. Service Consumer A and Service Consumer B access Service A's two service contracts (3, 4). Service Consumer C bypasses the service contracts and accesses the service logic directly (5).



You are told that the database and legacy system that are currently being used by Service A are being replaced with different products. The two service contracts are completely decoupled from the core service logic, but there is still a concern that the introduction of the new products will cause the core service logic to behave differently than before. What steps can be taken to change the Service A architecture in preparation for the introduction of the new products so that the impact on Service Consumers A, B, and C is minimized?

A. The Service Abstraction principle can be applied to hide the implementation details from the core service logic of Service A, thereby shielding this logic from changes to the implementation. In support of this, the Service Facade pattern can be applied to position Facade components between the core service logic and Service Consumers A and B. These Facade components will be designed to regulate the behavior of Service A . The Contract Centralization pattern can be applied to force Service Consumer C to access Service A via one of its existing service contracts.

B. A third service contract can be added together with the application of the Contract Centralization pattern. This will force Service Consumer C to access Service A via the new service contract. The Service Facade pattern can be applied to position a Facade component between the new service contract and Service Consumer C in order to regulate the behavior of Service A . The Service Abstraction principle can be applied to hide the implementation details of Service A so that no future service consumers are designed to access any of Service A's underlying resources directly.

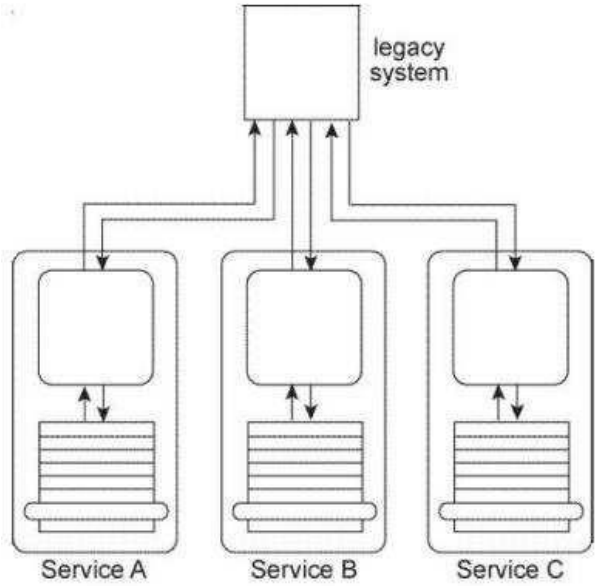
C. The Service Facade pattern can be applied to position Facade components between the core service logic and the two service contracts. These Facade components will be designed to regulate the behavior of Service A . The Contract Centralization pattern can also be applied to force Service Consumer C to access Service A via one of its existing service contracts. D. None of the above.

Correct Answer: C

Section: (none)
Explanation

Explanation/Reference:

QUESTION 13
Service A. Service B. and Service C are each designed to access the same shared legacy system. The service contracts for Service A, Service B, and Service C are standardized and decoupled from the underlying service logic. Service A and Service B are agnostic services that are frequently reused by different service compositions. Service C is a non-agnostic task service that requires access to the legacy system in order to retrieve business rules required for the service to make runtime decisions that determine its service composition logic. The legacy system uses a proprietary file format that Services A, B, and C need to convert to and from.



You are told that additional services need to be created, all of which need access to the legacy system. You are also told that the legacy system may be replaced in the near future. What steps can be taken to ensure that the replacement of the legacy system has a minimal impact on Services A, B, and C and any future services that are designed to rely upon it?

A. The Legacy Wrapper pattern can be applied together with the Standardized Service Contract principle to position a standardized service contract between the legacy system and any services that require access to it. This effectively establishes a new utility service dedicated to the encapsulation of the legacy system. When the legacy system is replaced, the utility service can keep its standardized service contract. To build the utility service, the Data Format Transformation pattern is applied to convert between the proprietary legacy system file format and the XML format used in the standardized service contract.

B. The Legacy Wrapper pattern can be applied together with the Official Endpoint pattern so that the Service A service contract is positioned as the sole access point for the legacy system. The Data Format Transformation pattern is applied to enable the conversion between the proprietary legacy system file format and the XML format used in the Service A service contract. Finally, the Contract Centralization pattern is applied so that Service A is forced to only access the legacy system via its published standardized service contract.

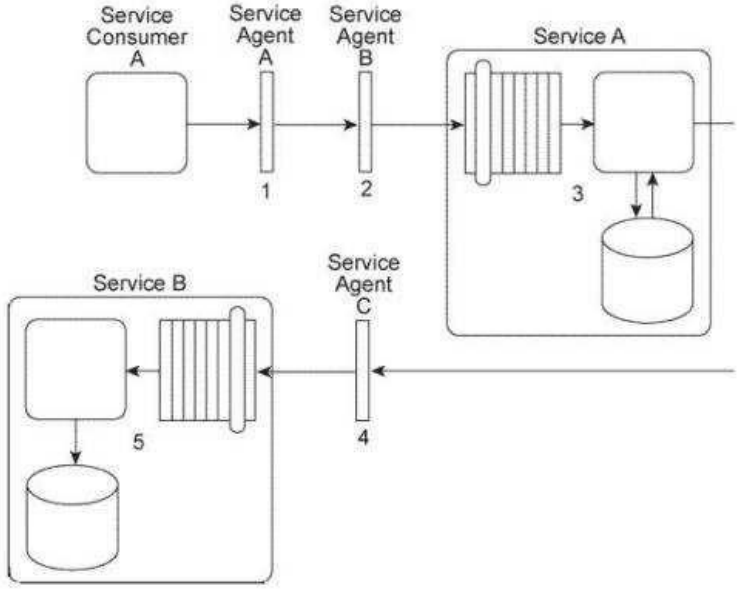
C. The Legacy Wrapper pattern can be applied together with the Data Format Transformation pattern and the Standardized Service Contract principle in order to establish an intermediate layer of standardized transformation logic that is positioned between the legacy system and Services A, B, and C. This way, if the legacy system is replaced, the services will not be affected because of the abstraction established by the standardized transformation layer. D. None of the above.

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:

QUESTION 14
Service Consumer A sends a message to Service A. Before the message arrives with Service A, it is intercepted by Service Agent A (1). which checks the message for compliance to Policy A that is required by Service A. If the message fails compliance, Service Agent A will not allow it to proceed and will instead write the message contents to a log. If the message does comply to the policy, it continues to be transmitted toward Service A, but before it arrives it is intercepted by Service Agent B (2), which validates the security credentials in the message header. If the security credential validation fails, the message is rejected and a runtime exception is raised. If the security credentials are validated, the message is sent to Service A.

Upon receiving the message, Service A retrieves a data value from a database and populates the message header with this data value (3) prior to forwarding the message to Service B. Before the message arrives at Service B. it is intercepted by Service Agent C (4) which checks the message for compliance with two policies: Policy B and Policy C. Policy B is identical to Policy A that was checked by Service Agent A. To check for compliance to Policy C. Service Agent C uses the data value added by Service A. If the message complies with both of the policies, it is forwarded to Service B (5), which stores the message contents in its own database.



You are told that Policy B and Policy C have changed. Also, in order to carry out the compliance check of Policy C, Service Agent C will now require a new data value from the Service B database. How can this service composition architecture be changed to fulfill these new requirements?

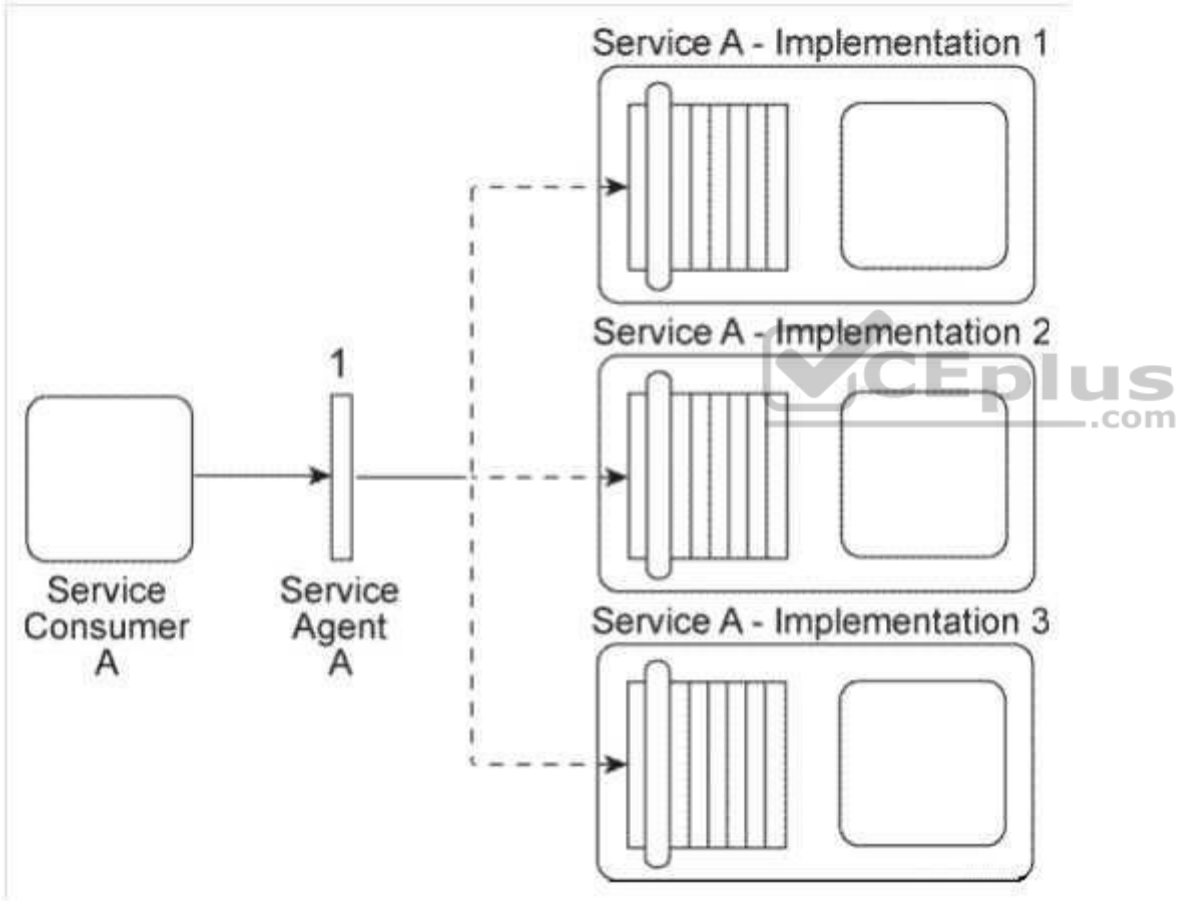
A. The Policy Centralization pattern can be applied so that only one service agent is used to enforce Policy A and Policy B. Service A is redesigned to first query Service B for the value required by Service Agent C to check the compliance of the updated Policy C. If the compliance check is successful, the message is sent to Service B .

B. The Policy Centralization pattern can be applied so that only one service agent is used to enforce Policy A and Policy B. Service Consumer A is redesigned to first query Service B for the value required by Service Agent C. This way, Service Consumer A can include this value in the message header prior to sending the message to Service A .

C. The Policy Centralization pattern can be applied so that only one service agent is used to enforce Policy A and Policy B. The policy enforcement logic for Policy C is removed from Service Agent C and instead embedded within the logic of Service B . This way, Service B can itself retrieve the value required to check compliance with Policy C. If the message received is not in compliance, Service B will reject it. D. None of the above.

Correct Answer: D
Section: (none)
Explanation

Explanation/Reference:
QUESTION 15
It has been confirmed that Policy A and Policy B are, in fact, the same policy and that the security credential check performed by Service Agent B also needs to be carried out on messages sent to Service B .



How can this service composition architecture be changed to reduce the redundancy of policy content and fulfill the new security requirement?

A. The Policy Centralization pattern can be applied so that Policy A and Policy B are combined into the same policy. The policy enforcement logic is removed from Service Agent C and Service Agent A is then used to enforce the policy for messages sent to Service A and Service B . Service Agent B can be used to perform the security credential check for Service A and Service B .
B. The Policy Centralization pattern can be applied so that Policy A and Policy B are combined into the same policy. The Service Agent pattern is then applied to introduce a new service agent (called Service Agent D) which carries out the validation and enforcement of Policy A and Policy B. Service Agent B can be moved so that it performs the security credential check for Service B, but not for Service A .
C. The Policy Centralization pattern can be applied so that Service Agent A is changed to enforce the policy for messages sent to Service A and Service B and to perform the security credential check for Service A and Service B . D. None of the above.

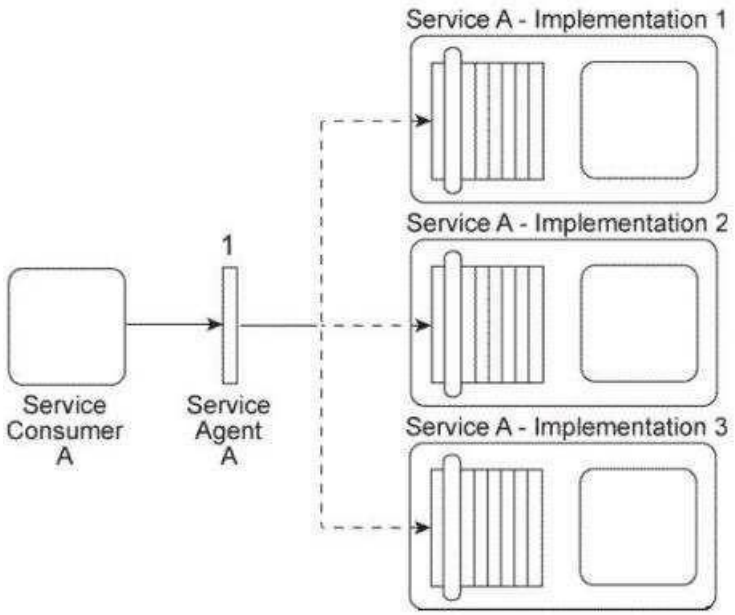
Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:

QUESTION 16
Service Consumer A sends a message to Service A. There are currently three duplicate implementations of Service A (Implementation 1, Implementation 2, Implementation 3).

The message sent by Service Consumer A is intercepted by Service Agent A (1), which determines at runtime which implementation of Service A to forward the message to.

All three implementations of Service A reside on the same physical server.



You are told that despite the fact that duplicate implementations of Service A exist, performance is still poor at times. Also, you are informed that a new service capability will soon need to be added to Service A that will introduce functionality that will require access to a shared database that is used by many other clients and applications in the IT enterprise. This is expected to add further performance demands on Service A . How can this service architecture be changed to improve performance in preparation for the addition of the new service capability?

A. The Standardized Service Contract principle is applied to ensure that the new service capability extends the existing service contract in a manner that is compliant with current design standards. The Redundant Implementation pattern is applied to establish separate implementations of Service A that include duplicate databases with copies of the data that Service A requires from the shared database.
B. The Service Autonomy principle is applied to further isolate the individual implementations of Service A by separating them onto different physical servers. When the new service capability is added, the Service Data Replication pattern is applied to give each implementation of Service A its own copy of the data it requires from the shared database.
C. The Service Loose Coupling principle is applied together with the Standardized Service Contract principle to ensure that Service Consumer A is not indirectly coupled to the shared database after the new service capability is added to the service contract. The Legacy Wrapper pattern can be applied to establish a new utility service that will provide standardized data access service capabilities for the shared database. D. None of the above.

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 17

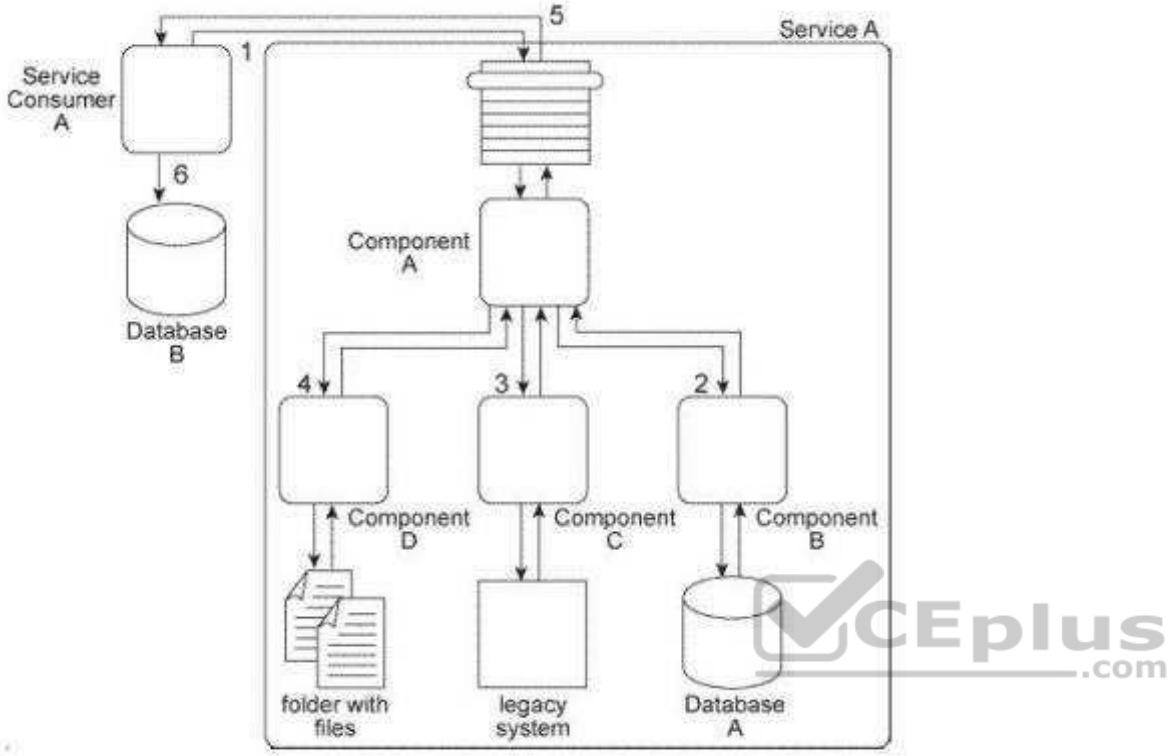
When Service A receives a message from Service Consumer A(1),the message is processed by Component A. This component first invokes Component B (2), which uses values from the message to query Database A in order to retrieve additional data. Component B then returns the additional data to Component A.

Component A then invokes Component C (3), which interacts with the API of a legacy system to retrieve a new data value. Component C then returns the data value back to Component A.

Next, Component A sends some of the data it has accumulated to Component D (4), which writes the data to a text file that is placed in a specific folder. Component D then waits until this file is imported into a different system via a regularly scheduled batch import. Upon completion of the import, Component D returns a success or failure code back to Component A.

Component A finally sends a response to Service Consumer A (5) containing all of the data collected so far and Service Consumer A writes all of the data to Database B (6).

Components A, B, C. and D belong to the Service A service architecture. Database A, the legacy system, and the file folders are shared resources within the IT enterprise.



Service A is a task service that completes an entire business task on its own without having to compose other services. However, you have received many complaints about the reliability of Service A . Specifically, it has three problems. First, when Component B accesses Database A, it may not receive a response for several minutes when the database is being accessed by other applications in the IT enterprise. Secondly, the legacy system accessed by Component C frequently crashes and therefore becomes unavailable for extended periods of time. Third, for Component D to respond to Component A, it must first wait for the batch import of the files to occur. This can take several minutes during which Service Consumer A remains stateful and consumes excessive memory. What steps can be taken to address these three problems?

A. The Legacy Wrapper pattern can be applied so that Component B is separated to wrap the shared database, thereby allowing Component A to interact with this new service instead of directly interacting with the database. The Legacy Wrapper pattern can be applied again so that Component C is separated into a separate service that acts as a wrapper of the legacy system API. Component D can then be separated into a separate service and the Event-Driven Messaging pattern can be applied to establish a publisher-subscriber relationship between this new service and Component A and between Service A and Service Consumer A. The interaction between Service Consumer A and Component A is then redesigned so that Component A issues a message back to Service Consumer A when the event related to the batch import is triggered.

B. The Service Data Replication pattern can be applied so that Component B can access a replicated database instead of having to access the shared Database A directly. The Legacy Wrapper pattern can be applied so that Component C is separated into a separate service that acts as a wrapper of the legacy system API. Next, the Reliable Messaging pattern can be applied so that acknowledgements are issued from the new wrapper service to Component A, thereby enabling notifying Component A during times when the legacy system is unavailable. Finally, Component D is separated into a separate service and the Event-Driven
Messaging pattern is applied to establish a publisher-subscriber relationship between this new service and Component A. The interaction between Service Consumer A and Component A is then redesigned so that Component A first interacts with Component B and the new wrapper service. Service A then issues a final message back to Service Consumer A.

C. The Service Data Replication pattern can be applied so that Component B can access a replicated database instead of having to access the shared Database A directly. The Legacy Wrapper pattern can be applied so that Component C is separated into a separate service that acts as a wrapper of the legacy system API. Next, the Asynchronous Queuing pattern can be applied so that a messaging queue is positioned between Component A and the new wrapper service, thereby enabling communication during times when the legacy system is unavailable. Finally, Component D is separated into a new service and the Event-Driven
Messaging pattern is applied to establish a publisher-subscriber relationship between this service and Component A and between Service A and Service Consumer A. The interaction logic is redesigned as follows: Component A interacts with Component B, the new wrapper service, and then issues a request to the new event-driven service. Upon receiving a response triggered by the event related to the batch import, Service A responds to Service Consumer A. D. None of the above.

Correct Answer: C
Section: (none)
Explanation

Explanation/Reference:

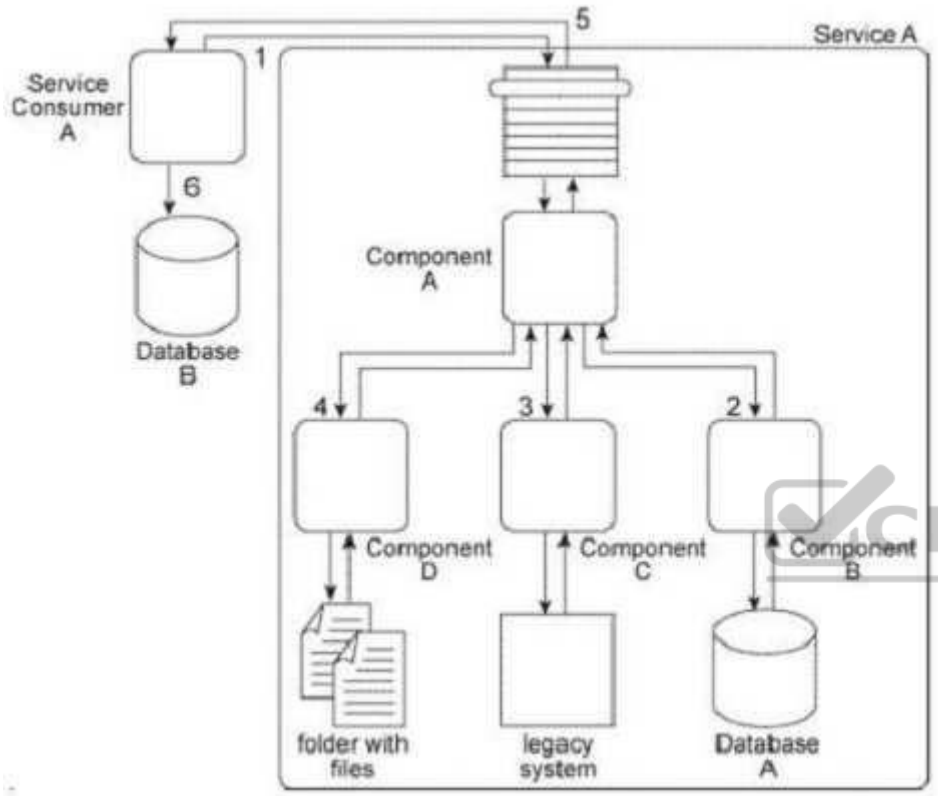
QUESTION 18
When Service A receives a message from Service Consumer A(1),the message is processed by Component A. This component first invokes Component B (2), which uses values from the message to query Database A in order to retrieve additional data. Component B then returns the additional data to Component A.

Component A then invokes Component C (3), which interacts with the API of a legacy system to retrieve a new data value. Component C then returns the data value back to Component A.

Next, Component A sends some of the data it has accumulated to Component D (4), which writes the data to a te>X file that is placed in a specific folder. Component D then waits until this file is imported into a different system via a regularly scheduled batch import. Upon completion of the import, Component D returns a success or failure code back to Component A.

Component A finally sends a response to Service Consumer A (5) containing all of the data collected so far and Service Consumer A writes all of the data to Database B (6).

Components A, B, C. and D belong to the Service A service architecture. Database A, the legacy system, and the file folders are shared resources within the IT enterprise.



Service A is an entity service with a service architecture that has grown over the past few years. As a result of a service inventory-wide redesign project, you are asked to revisit the Service A service architecture in order to separate the logic provided by Components B, C, and D into three different utility services without disrupting the behavior of Service A as it relates to Service Consumer A . What steps can be taken to fulfill these requirements?

A. The Legacy Wrapper pattern can be applied so that Component B is separated into a separate wrapper utility service that wraps the shared database. The Asynchronous Queuing pattern can be applied so that a messaging queue is positioned between Component A and Component C, thereby enabling communication during times when the legacy system may be unavailable or heavily accessed by other parts of the IT enterprise. The Service Facade pattern can be applied so that a Facade component is added between Component A and Component D so that any change in behavior can be compensated. The

Service Autonomy principle can be further applied to Service A to help make up for any performance loss that may result from splitting the component into a separate wrapper utility service.

B. The Legacy Wrapper pattern can be applied so that Component B is separated into a separate utility service that wraps the shared database. The Legacy Wrapper pattern can be applied again so that Component C is separated into a separate utility service that acts as a wrapper for the legacy system API. The Legacy Wrapper pattern can be applied once more to Component D so that it is separated into another utility service that provides standardized access to the file folder. The Service Facade pattern can be applied so that three Facade components are added: one between Component A and each of the new wrapper utility services. This way, the Facade components can compensate for any change in behavior that may occur as a result of the separation. The Service Composability principle can be further applied to Service A and the three new wrapper utility services so that all four services are optimized for participation in the new service composition. This will help make up for any performance loss that may result from splitting the three components into separate services.

C. The Legacy Wrapper pattern can be applied so that Component B is separated into a separate utility service that wraps the shared database. The Legacy Wrapper pattern can be applied again so that Component C is separated into a separate utility service that acts as a wrapper for the legacy system API. Component D is separated into a separate service and the Event-Driven Messaging pattern is applied to establish a publisher-subscriber relationship between this new service and Component A. The interaction between Service Consumer A and Component A is then redesigned so that Component A first interacts with Component B and the new wrapper service. Service A then issues a final message back to Service Consumer A. The Service Composability principle can be further applied to Service A and the three new wrapper utility services so that all four services are optimized for participation in the new service composition. This will help make up for any performance loss that may result from splitting the three components into separate services. D. None of the above.

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

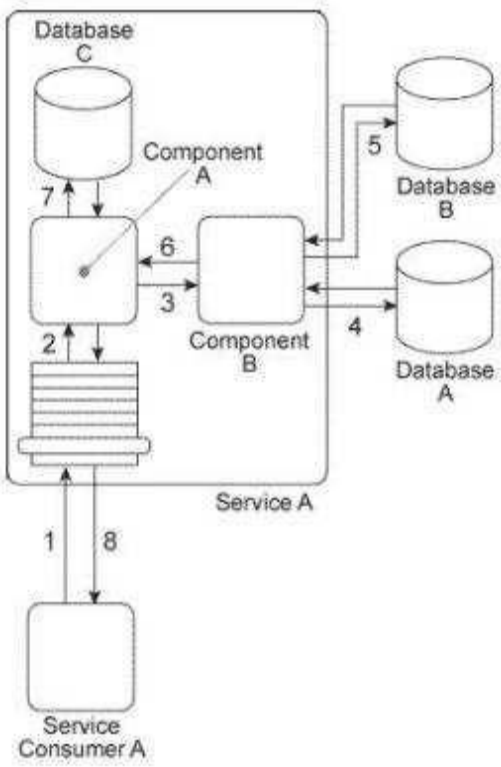
QUESTION 19

Service Consumer A sends Service A a message containing a business document (1). The business document is received by Component A, which keeps the business document in memory and forwards a copy to Component B (3). Component B first writes portions of the business document to Database A (4).

Component B writes the entire business document to Database B and then uses some of the data values from the business document as query parameters to retrieve new data from Database B (5).

Next, Component B returns the new data back to Component A (6), which merges it together with the original business document it has been keeping in memory and then writes the combined data to Database C (7). The Service A service capability invoked by Service Consumer A requires a synchronous request-response data exchange. Therefore, based on the outcome of the last database update, Service A returns a message with a success or failure code back to Service Consumer A (8).

Databases A and B are shared and Database C is dedicated to the Service A service architecture.



There are several problems with this architecture: First, the response time of Database A is often poor, resulting in Component B taking too much time to provide a response to Component A. This results in Component A consuming too many runtime resources while it holds the business document in memory and it also causes unreasonable delays in responding to Service Consumer A. Additionally, Database B is being replaced with a different database product that supports a proprietary file format. This will disable the current interaction between Component B and the new Database B. What steps can be taken to solve these problems?

A. The State Repository pattern is applied so that Component A can defer the business document data to a state database while it waits for a response from Component B. The Service Data Replication pattern is applied so that Component B can interact with a database that is replicated from the shared Database A.
This will improve performance and reliability that will affect both Component A and Service Consumer A. Finally, the Legacy Wrapper pattern is applied so that Database B is wrapped in a standardized contract. This will establish a new wrapper utility service that will allow Database B to be replaced with a different database product without affecting Service A . Furthermore, the Data Format Transformation pattern can be applied within the new wrapper utility service to enable it to convert to and from the new proprietary file format.

B. The State Repository pattern is applied so that Component A can defer the business document data to a state database while it waits for a response from Component B. The Asynchronous Queuing pattern can be applied so that a messaging queue is established between Service Consumer A and Service A,

thereby guaranteeing delivery and avoiding Service Consumer A from being tied up too long waiting for Service A to respond. Finally, the Data Format Transformation pattern can be applied to enable Component B to convert to and from the new proprietary file format introduced by the database product that is replacing Database B.

C. The Legacy Wrapper pattern is applied so that Database B is wrapped in a standardized contract. This will establish a new wrapper utility service that will allow Database B to be replaced with a different database product without affecting Service A . The Data Format Transformation pattern can be applied within the new wrapper utility service to enable it to convert to and from the new proprietary file format. The Service Data Replication pattern is applied so that Component B can interact with a database that is replicated from the shared Database B, regardless of what database product is used to replace Database B. The Service Abstraction principle can be further applied to hide the implementation details, including the changes mentioned in this solution, from Service Consumer A. D. None of the above.

Correct Answer: A
Section: (none)
Explanation

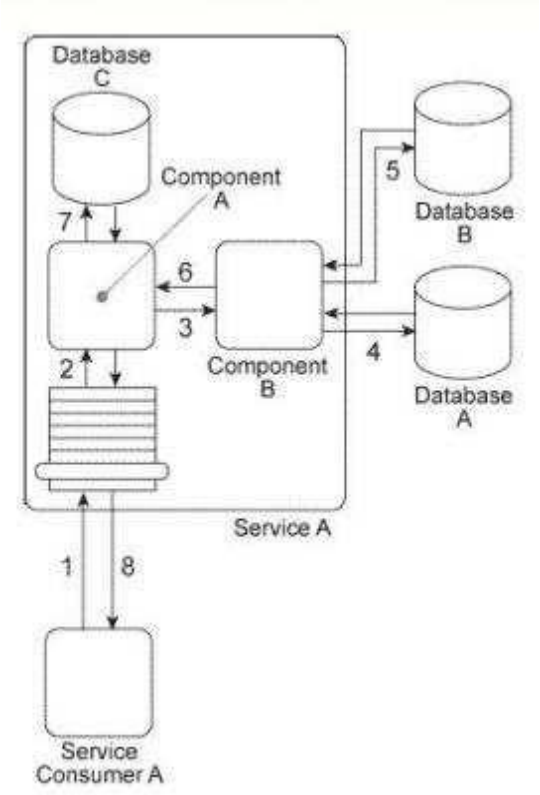
Explanation/Reference:

QUESTION 20
Service Consumer A sends Service A a message containing a business document (1). The business document is received by Component A, which keeps the business document in memory and forwards a copy to Component B (3). Component B first writes portions of the business document to Database A (4).

Component B writes the entire business document to Database B and then uses some of the data values from the business document as query parameters to retrieve new data from Database B (5).

Next, Component B returns the new data back to Component A (6), which merges it together with the original business document it has been keeping in memory and then writes the combined data to Database C (7). The Service A service capability invoked by Service Consumer A requires a synchronous request-response data exchange. Therefore, based on the outcome of the last database update, Service A returns a message with a success or failure code back to Service Consumer A (8).

Databases A and B are shared and Database C is dedicated to the Service A service architecture.



There are several problems with this architecture: The business document that Component A is required to keep in memory (while it waits for Component B to complete its processing) can be very large. Especially when Service A is concurrently invoked by multiple service consumers, the amount of runtime resources it uses to keep this data in memory can decrease the overall performance of all service instances. Additionally, because Database A is a shared database that sometimes takes a long time to respond to Component B, Service A can take a long time to respond back to Service Consumer A . Currently, Service Consumer A will wait for a response for up to 30 seconds after which it will assume the request to Service A has failed and any subsequent response messages from Service A will be rejected. What steps can be taken to solve these problems?

A. The Service Statelessness principle can be applied together with the State Repository pattern in order to extend Database C so that it also becomes a state database allowing Component A to temporarily defer the business document data while it waits for a response from Component B. The Service Autonomy principle is applied together with the Legacy Wrapper pattern to isolate Database A so that it is encapsulated by a separate wrapper utility service. The Compensating Service Transaction pattern is applied so that if the response time of Service A exceeds 30 seconds, a notification is sent to a human administrator to raise awareness of the fact that the eventual response of Service A will be rejected by Service Consumer A.

B. The Service Statelessness principle can be applied together with the State Repository pattern in order to establish a state database that Component A can defer the business document data to while it waits for a response from Component B. The Service Autonomy principle can be applied together with the Service Data Replication pattern to establish a dedicated replicated database for Component B to access instead of the shared Database A. The Asynchronous Queuing pattern can be applied to establish a messaging queue between Service Consumer A and Service A so that Service Consumer A does not need to remain stateful while it waits for a response from Service A .

C. The Service Statelessness principle can be applied together with the State Repository pattern in order to establish a state database that Component A can defer the business document data to while it waits for a response from Component B. The Service Autonomy principle can be applied together with Service Abstraction principle, the Legacy Wrapper pattern, and the Service Facade pattern in order to isolate Database A so that it is encapsulated by a separate wrapper utility service and to hide the Database A implementation from Service A and to position a Facade component between Component B and the new wrapper service. This Facade component will be responsible for compensating the unpredictable behavior of Database A. D. None of the above.

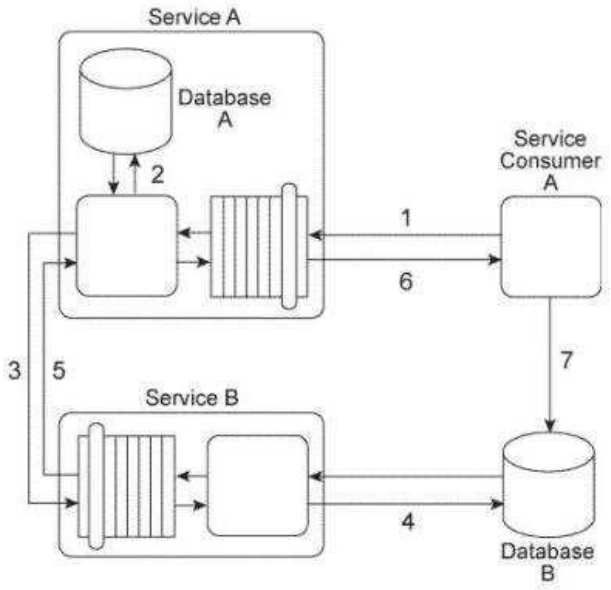
Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

QUESTION 21
Service Consumer A sends a message with a business document to Service A (1), which writes the business document to Database A (2). Service A then forwards the business document to Service B (3), which writes the business document to Database B (4).

Service B then responds to Service A with a message containing a failure or success code (5) after which Service A responds to Service Consumer A with a message containing a failure or success code (6). Upon receiving the message, Service Consumer A updates a log table in Database B (7). The log entry is comprised of the entire business document.

Database A is dedicated to the Service A service architecture and Database B is a shared database.



You are told that the database updates performed by Service A and Service B must be either both successful or they cannot happen at all. The database update performed by Service Consumer A must happen after it is given the outcome of the database updates performed by Service A and Service B . Given that Service Consumer A must also update Database B as part of this service composition architecture, how is it possible to fulfill these requirements?

A. The State Repository pattern can be applied so that Service A writes the business document data to a separate state database until it receives a response message from Service B . If the response message contains a success code, Service A writes the business document to Database A. If the response contains a failure code, Service A discards the data that was written to the state database.
B. The Service Data Replication pattern can be applied to Service Consumer A and Service B so that separate dedicated databases can be established allowing Service Consumer A to make updates independently of Service B . Service A is simply redesigned to not write the business document to Database A until after it receives a message containing a success code from Service B.
C. The Atomic Service Transaction pattern can be applied to encompass Service A, Service B and Service Consumer A. This will guarantee that all of the actions performed by the service composition participants will either be successful or will be rolled back if anyone is not successful. D. None of the above.

Correct Answer: D
Section: (none)
Explanation

Explanation/Reference: